Project 2.2: Study of a 2D random walk

Superior School of Physics and Mathematics, Mexico City, Mexico.

Numerical Physics

Cruz Perez Gustavo Hazael

Email: haxaleg@gmail.com

1. Equations solved.

One of the applications of the generation of random numbers is the simulation of random walks, so far we have the theory of how to generate a number $N$of random numbers, if we generate two independent lists, we can consider them as the lengths of steps on each axis of a 2D random walk.

$$(\Delta x_1, \Delta y_1), (\Delta x_2, \Delta y_2) \ \ldots . (\Delta x_{N-1}, \Delta y_{N-1})(\Delta x_N, \Delta y_N)$$

At the end of this number of steps the radial distance that the walker moved, we can obtain it from the following equation.

$$R^2(N) = (\Delta x_1 + \Delta x_2 + \cdots + \Delta x_{N-1} + \Delta x_N)^2 + (\Delta y_1 + \Delta y_2 + \cdots + \Delta y_{N-1} + \Delta y_N)^2 \qquad \ldots (1)$$

$$R_{rms} = \sqrt{(\Delta x_1 + \Delta x_2 + \cdots + \Delta x_{N-1} + \Delta x_N)^2 + (\Delta y_1 + \Delta y_2 + \cdots + \Delta y_{N-1} + \Delta y_N)^2}$$

We can get a simpler expression for , because if we develop.R_{rms}

$$R^2(N) = (\Delta x_1)^2 + \cdots + (\Delta x_N)^2 + (\Delta y_1)^2 + \cdots + (\Delta y_N)^2 + \sum_{i \neq j}^{N} \sum_{j \neq i}^{N} \Delta x_i \Delta x_j + \sum_{i \neq j}^{N} \sum_{j \neq i}^{N} \Delta y_i \Delta y_j$$

In addition, it was hypothesized that the walk is random and each step given has the same probability of occurrence in all directions, so if we take the expected value, for a large number of steps, we can obtain the mean square value of $\langle \Delta x_i \Delta x_j \rangle = 0R$.

$$R_{rms}^2 \approx \langle R^2(N) \rangle = \left\langle (\Delta x_1)^2 + \cdots + (\Delta x_N)^2 + (\Delta y_1)^2 + \cdots + (\Delta y_N)^2 + \sum_{i \neq j}^{N} \sum_{j \neq i}^{N} \Delta x_i \Delta x_j + \sum_{i \neq j}^{N} \sum_{j \neq i}^{N} \Delta y_i \Delta y_j \right\rangle$$

$$R_{rms}^2 = \langle (\Delta x_1)^2 + \cdots + (\Delta x_N)^2 + (\Delta y_1)^2 + \cdots + (\Delta y_N)^2 \rangle$$

$$R_{rms}^2 = \langle (\Delta x_1)^2 + (\Delta y_1)^2 \rangle + \cdots + \langle (\Delta x_1)^2 + (\Delta y_1)^2 \rangle$$

$$R_{rms}^2 = N\langle r^2 \rangle = N r_{rms}^2$$

$$R_{rms} = \sqrt{N} r_{rms} \quad \ldots (2)$$

2. Numerical method or algorithm used.

To generate a random walk of Nsteps, python used the random library, with which we could fill two arrays of $X\ e\ Y$ random numbers in an interval of 0 to 1.

$$X = (\Delta x'_1, \ldots, \Delta x'_N)$$

$$Y = (\Delta y'_1, \ldots, \Delta y'_N)$$

$$(\Delta x'_1, \Delta y'_1), (\Delta x'_2, \Delta y'_2), \ldots, (\Delta x'_{N-1}, \Delta y'_{N-1}), (\Delta x'_N, \Delta y'_N)$$

We normalize each coordinate so that the distance of each step is 1.

$$\Delta x_i = \frac{\Delta x'_i}{L} \quad ; \quad \Delta y_i = \frac{\Delta y'_i}{L}$$

$$L = \sqrt{\Delta x_i'^2 + \Delta y_i'^2}$$

Where it must be fulfilled.

$$\frac{\langle \Delta x_i \Delta x_{j \neq y} \rangle}{R^2(N)} \approx 0 \quad \ldots (3)$$

For a given number N of steps the random walk k is repeated times where , each of these k walks will have N steps starting each with different seed. If we calculate for each one we can obtain its expected value, with the average of the k experiments.$k = \sqrt{N} R^2(N)$

$$R^2(N) = (\Delta x_1 + \cdots + \Delta x_N)^2 + (\Delta y_1 + \cdots + \Delta y_N)^2$$

$$\langle R^2(N) \rangle = \frac{1}{k} \sum_{i=1}^{k} R_i^2(N)$$

And get at the end a value of given by.$R_{rms}(N)$

$$R_{rms}(N) = \sqrt{\langle R^2(N) \rangle} = \sqrt{\frac{1}{k} \sum_{i=1}^{k} R_i^2(N)} \quad \ldots (4)$$

3. Code listing.

Table 1: List of code that resolves subparagraphs a), b), c) and e)

```
import matplotlib.pyplot as pl
from pylab import *
import numpy as np
from math import*

n=int(input("Type the number of N steps: "))
k=int(sqrt(n))

all=[]

# subparagraphs a), b), c)
large def(n,all):
    Rms=[]

    #inciso (a)
    def chameleat():
        sumx=0.0
        sumy=0.0

        x = np.zeros(n)
        y = np.zeros(n)
        for i in range(1,n):#genera normalized numbers
            x[i]=(np.random.rand()-0.5)* 2.    # -1 =< x =< 1
            y[i]=(np.random.rand()-0.5)* 2.    # -1 =< y =< 1
            L=sqrt(x[i]**2 + y[i]**2)
            x[i]=x[i]/L
            y[i]=y[i]/L

        for i in range(1,n):
            sumx+=x[i]
            sumy+=y[i]

        Rms.append(sumx**2+sumy**2)
    chameleat()


    #inciso (b)
    def kexperimentos():
        for i in range(1,k):
            np.random.seed(None)
            chameleat()
    kexperiments()

    #inciso c) obtain the expected value for k hikes
    def vespR(Rms,n):
        Total=0.0
        for i in range(k):
            Total+=Rms[i]

        Total=Total/k
        all.append(Total)
        print(f'THE value of <R^2({n})> is: {Total}')
    vespR(Rms,n)
for i in range(n+1):
```

```
    big(i,everything)


#inciso (e)
Def Graph(All):
    N=[]
    for i in range(n+1):
        N.append(sqrt(i))
        all[i]=sqrt(everything[i])

    lin=[0,N[n-1]]
    pl.title(f"Rrms in root function of N")
    pl.plot(N,all,'b',linewidth=0.8)
    pl.plot(N,N,'g')
    pl.xlabel("sqrt(N)")
    pl.ylabel("Rrms")
    pl.show()
Graphics(all)
```

Table 2: Code that verifies the theoretical hypothesis (subsection d)) and graphs the route of a walk of N steps entered by the user

```
#Este Program generates the graph for a number N of steps entered by the user
#y the theoretical hypothesis subsection (d) is verified
import numpy as np
import matplotlib.pylab as pl
Import random
from math import*


n=int(input("Enter the number of steps of the walk: "))

x = np.zeros(n)
y = np.zeros(n)
sx=np.zeros(n)
sy=np.zeros(n)

for i in range(1,n):#genera normalized numbers
x[i]=(np.random.rand()-0.5)* 2.    # −1 =< x =< 1
y[i]=(np.random.rand()-0.5)* 2.    # −1 =< y =< 1
L=sqrt(x[i]**2 + y[i]**2)
x[i]=x[i]/L
y[i]=y[i]/L

sx[i]=x[i]+sx[i-1]
sy[i]=y[i]+sy[i-1]

R2=sx[n-1]**2 + sy[n-1]**2

pl.plot(sx,sy,'k',linewidth=0.7);p l.plot(0,0,'ro')
pl.grid(True)
pl.title(f'Walk with {n} steps')
pl.xlabel("x")
pl.ylabel("y")
pl.show()

def hypothesis(x,y):
```

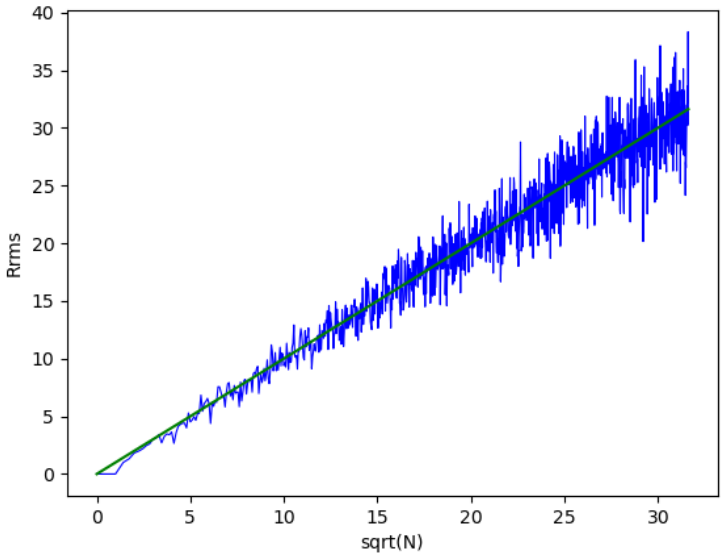
```
htl=[]
for i in range(n):
for j in range(n):
if i==j or i>j:
Pass
else:
ht=(x[i]*x[j])
htl.append(ht)

sum=0
for i in range(len(htl)):
sum=sum+htl[i]

hypothet=sum/(len(htl)*R2)
print(f"\nThe theoretical hypothesis is: {hypothetic}")
Hypothesis(x,y)
```

4. Visualization.

The code to solve this problem was necessary to divide it into two parts, table 1 shows the code that generates normalized random walks and for each value of N is sent to call the function that generates k-walks for a given N, also calculates the value of , for several values of N.$\langle R^2(N)\rangle$

Table 3: Different expected values of $\langle R^2(N)\rangle$

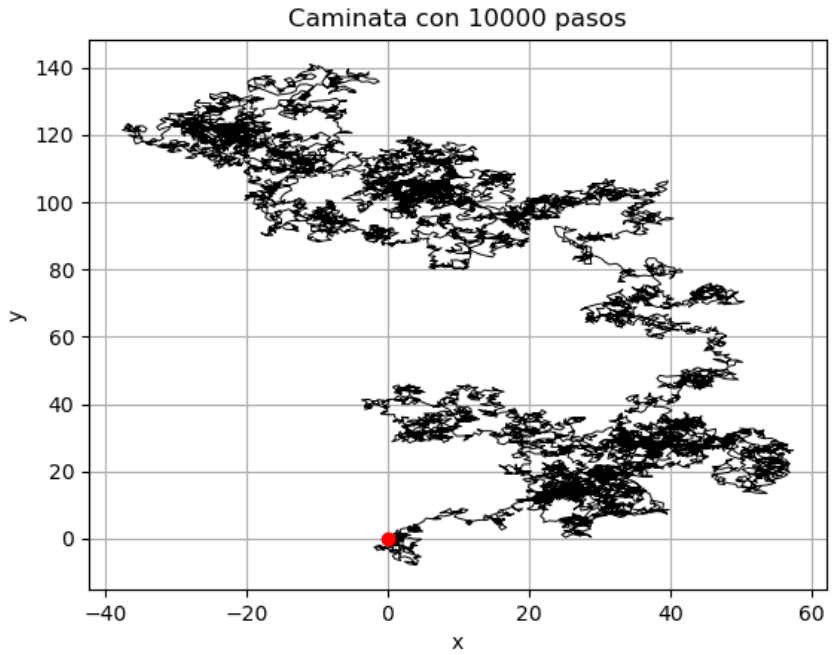
N	K	$\langle R^2(N)\rangle$
1000	31	867.2732345638726
900	30	857.5038413351655
800	28	669.0126375307456
700	26	732.5727053058826
600	24	651.0382859864789
500	22	410.94515024951494
400	22	383.8836181569367



Graph 1: In blue the behavior of Rrms as a function of the root of N, in green $\sqrt{N}$ *vs* $\sqrt{N}$

The second part of the code generates a single walk, with N steps entered by the user and makes the route graph, also here the theoretical hypothesis is verified where the following value is obtained for $N = 1000$

$$\frac{\langle \Delta x_i \Delta x_{j \neq i} \rangle}{R^2} = -1.41506183661474 \times 10^{-6} \approx 0$$



Graph 2: Random walk generated by the algorithm, for 10,000 steps that starts from the origin (red dot)

5. **Discussion.**

We see in table 3 that the expected value $\langle R^2(N)\rangle$ obtained from averaging the k experiments for a given N, is approximately N this was to be expected then.

$$\langle R^2(N)\rangle = N\langle r^2\rangle$$

And when doing the normalization of each of the steps we have to, so theoretically we have.$\langle r^2\rangle$

$$\langle R^2(N)\rangle = N$$

Taking the square root.

$$\sqrt{\langle R^2(N)\rangle} = \sqrt{N}$$

$$R_{rms} = \sqrt{N}$$

Hence, graph 2 behaves like the identity when plotted as a function of , we can also observe a dispersion of , as the value of $R_{rms}\sqrt{N}R_{rms}\sqrt{N}$

6. **Criticizes.**

We can find certain deficiencies in the implemented code, such as when checking the theoretical hypothesis equation (3), you have a doubly nested for, which in asymptotic notation have one of the worst complexities, which makes that for very large values of N the computer takes a long time to$O(n^2)$ calculate its value, however it is not necessary to get to check it with excessively large numbers, because it has been proven that it $R^2(N)$ grows as a function of N therefore (3) decreases as N increases.

The importance of the study of random walks have a lot of applications in different areas of science so a refinement of the theory and computational methods that generate these simulations such as the Metropolis-Hastings rhythm is needed, in our case the next step in a physical application, is the Brownian movement of a particle in a fluid, where we could generate a random walk in 3D and that the direction taken by the particle was not completely random, that is, we can program some of the properties of the liquid that is being modeled, and that affect the direction of the particle from which its movement is studied, all this would need a model from more complex EDP.

7. **Bibliography.**

1) Rubin H. Landau, Manuel Paez, Cristian Constantin Bordeianu. (2012). Computational Physics Problem Solving with Computers. United States of : Wiley-VCH.